Gymnázium Nad Štolou 1, Praha 7

Maturitní Práce

Jednoduchá počítačová hra

Autor práce: Vojtěch Hána
Vedoucí práce: Martin Sourada
Třída: 5S

2021/2022

Prohlášení: Prohlašuji, že jsem maturitní práci vypracoval samostatně, použil jsem pouze podklady uvedené v přiloženém seznamu a postup při zpracování a dalším nakládání s prací je v souladu se zákonem č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů.

V Praze dne 8. března 2023

Zde poděkování

Obsah

Úvod **6**

I Teoretická část 7

1 Užité nástroje **7**

1.1 Engine 7

1.2 Další užité nástroje 7

1.2.1 GIMP 7

1.2.2 Audacity 8

1.2.3 Git 8

1.2.4 Blender 8

2 Vývoj **10**

2.1 Počáteční cíle 10

2.2 Průběh vývoje hry 10

2.2.1 Build 0.0.1 10

2.2.2 Build 0.0.3 11

2.2.3 Build 0.0.4 11

2.2.4 Neexistující build 0.0.5 a přechod na 2D 11

2.2.5 Build 0.1.0 12

2.3 Algoritmus AI počítačových protivníků 12

2.3.1 MasterSpawner 13

II Uživatelská příručka 15

1 Cíl hry **15**

2 Ovládání **15**

Závěr **16**

Resumé **17**

Příloha **18**

Seznam zdrojů a referencí **20**

Úvod

Lorem ipsum

Část I

Teoretická část

1 Užité nástroje

1.1 Engine

Veřejnosti je přístupno nepřeberné množství herních enginů, každý z nich se svými plusy a mínusy, orientovaný na jiný segment trhu. Například *Unigine* exceluje ve velkých scénách díky 64bitovým souřadnicím, zatímco *Clausewitz* je zaměřený na top-down grand-strategy hry.
Jelikož Astra je mým prvním velkým projektem, co se vývinu her týče, bylo pro mne důležité, aby pro zvolený engine byla dostupná rozsáhlá a kvalitní dokumentace a aktivní komunita, na kterou by se bylo možná obrátit v případě potíží. Tím z výběru vypadávají všechny málo známé enginy a také ty, které sice známé jsou, ale většinou na nich vyvíjejí pouze velká studia, jako například *CryEngine*.
Po tomto prvním kroku zbývali dva seriózní kandidáti, a to *Unity* a *Unreal Engine*. Finální volba nakonec padla na novější Unreal Engine 5 (nadále v této práci zkracován UE). Ten se jevil jako lepší volba díky jeho systémům jako je například engine osvětlení Lumen, který by mi značně usnadnil tvorbu realisticky vypadající grafiky.

1.2 Další užité nástroje

UE má ve své základní distribuci zabudováno rozsáhlé množství nástrojů pro tvorbu a nakládání s různými soubory. I přesto bylo v průběhu vývoje nutné využít několik dalších programů:

1.2.1 GIMP

GNU Image Manipulation Program (GIMP) je bezplatný rastrový grafický editor, který slouží k úpravě a tvorbě obrázků. Tento program jsem využil na tvorbu některých textur a design prvků uživatelského rozhraní.
Oproti konkurenčním programům, např. Adobe Photoshop, jsem GIMP zvolil zejména kvůli

předchozí znalosti tohoto programu.

1.2.2 Audacity

Audacity je open-source program pro úpravu zvukových souborů. Vzhledem k možnostem UE základně upravovat zvukové stopy (hlasitost, výška, filtry) přímo v enginu jsem Audacity užil jen zřídka a to na úpravu hlasových stop pro postavu hráčovy AI pomocnice a pro přípravu dalších zvukových stop převodem na podporovaný kodek.

1.2.3 Git

Git je distribuovaný systém správy verzí, který se používá pro sledování změn v kódu a koordinaci práce více lidí na společném projektu. Je to nástroj, který umožňuje uživatelům ukládat, verzovat a spravovat svůj kód a jeho historii.
Git umožňuje uživatelům pracovat na kopii kódu, kterou si mohou upravovat bez toho, aby se to projevilo na hlavní větvi (tzv. ”master branch”). Poté, co provedou své změny a jsou s nimi spokojeni, mohou je zahrnout do hlavní větve. Git také umožňuje ukládat různé verze kódu v různých větvích, což usnadňuje vývoj a testování nových funkcí nebo náprav chyb. Díky němu je také snadnější zpětné získání předchozích verzí kódu a práce na nich.
Jelikož jsem na projektu pracoval sám, řady z vlastností Gitu jsem vůbec nevyužil. Stejně tak zůstala nevyužitá i možnost vývoje ve více větvích (z důvodu, že jsem dříve nebyl se systémy sledování verzí), což není optimální.

1.2.4 Blender

Blender je otevřený a zdarma dostupný software pro 3D modelování, animaci a vizualizaci, široce používaný v oblasti filmového a herního průmyslu, architektury, designu a mnoha dalších. Uživatelům umožňuje vytvářet 3D objekty, animovat je, texturovat, osvětlovat a renderovat. Další funkcí Blenderu jsou například simulace fyziky, jako jsou srážky a deformace, simulace tekutin a plamenů apod.
Uživatelé také mohou vytvářet složité animace pomocí animačních klíčů, sledování kamer a světel, a dalších funkcí. Právě tuto konkrétní funkci jsem při vývoji použil, a to na vytvoření krátkého videa s logem hry, které se přehrává, když je hra spuštěna.

2 Vývoj

2.1 Počáteční cíle

Celkový koncept hry se během vývoje několikrát změnil. Původní plánování jsem ale provedl s následujícími cíli:

- Střílečka z pohledu třetí osoby (*Third-person shooter*), zasazený ve vesmírném prostředí.
- Pohybový model hráčovy vesmírné loďe, který umožňuje pohyb po všech šesti osách (tj. 3 osy trojrozměrného prostoru + 3 osy otáčení)
- Absence konečného úkolu – hra skončí pouze selháním či přáním hráče
- Počítačoví protivníci, kteří se dokáží v prostoru pohybovat stejným způsobem jako hráč a jsou schopni na něj efektivně útočit
- Realisticky vypadající (nestylizovaná) grafika

Za inspiraci měli sloužit podobné hry, jako například nedávno vydané *Star Wars: Squadrons* a starší *Star Wars: Battlefront II* od Electronic Arts nebo *Elite: Dangerous* od Frontier Developments.

2.2 Průběh vývoje hry

UE uživatelům nabízí několik předem zhotovených šablon, které obsahují všechny základní nezbytné prvky a dovolují uživatelům pouze rozšiřovat z tohoto funkčního základu. Této možnosti jsem nevyužil, jelikož žádná z nabízených šablon nevyhovovala výše uvedeným požadavkům. Zejména problematická by byla implementace trojrozměrného pohybu – ve všech šablonách byl pohyb dvourozměrný (pomineme-li např. skákání)

2.2.1 Build 0.0.1

Prvním úkolem pak tedy bylo vytvořit prázdnou úroveň a naprogramovat základní funkce. Původní letecký model z této doby byl ovládaný kombinací myši a klávesnice. Klávesnice zajišťovala rotaci okolo vektoru pohybu, zatímco myš zajišťovala stáčení. Loď cestovala neměnnou rychlostí (pokud nedošlo ke kolizi). V úrovni se taky vyskytoval jeden „nepřítel“, který sloužil jako test systému střelby. Nepřítel samotný se nijak nepohyboval ani neútočil.

Jediná jeho implementovaná funkce bylo jeho zničení, kdy po určitém počtu zásahů zmizel.

2.2.2 Build 0.0.3

Tato verze zaznamenala několik zlepšení k funkcím hráčovy lodě. Nově hráč mohl změnit rychlost letu v rámci povoleného rozsahu. Bohužel tato funkce kvůli chybě v programu se ve finální zkompilované hře nezobrazují prvky uživatelského rozhraní, které rychlost ukazují, a tak je občas změnu rychlostí obtížné postřehnout. Dále bylo odstraněno několik chyb, které způsobovaly, že loď střílela jinam, než ukazoval zaměřovač na obrazovce (který rovněž ve zkompilované hře není přítomen). V neposlední řadě došlo ke změnění osvětlení úrovně a přidání několika překážek v podobě asteroidů.

2.2.3 Build 0.0.4

Tato verze byla prozatím poslední, ve které jsem se plánoval zabývat lodí hráče. Byla zvýšena rychlost zatáčení lodě i její akcelerace. Debugové přímky při střelbě ze zbraní byly nahrazeny hezčími lasery, které mimo jiné vydávají slabé světlo. Byl také zpraven bug, který nedovolil zobrazení HUDu ve zkompilované hře. Hra také v této verzi dostala současné jméno Astra.
Hráče jsem tímto považoval za hotového a další položkou na seznamou byla implementace nepřátel.

2.2.4 Neexistující build 0.0.5 a přechod na 2D

První vlastností, kterou jsem chtěl nepřátelům implementovat, byl pohyb. Bohužel, pathfinding v trojrozměrném prostoru se z důvodů blíže popsaných v kapitole o algoritmu nepřátel ukázal jako extrémně obtížný na implementaci, alespoň pro začátečníka. Došel jsem tedy k rozhodnutí zcela předělat koncept hry na top-down (tj. se statickou kamerou zhlížející na hru zezhora.) vertikálně skrolující shoot-‘em-up hra.
V takových hrách se většinou nepřátelé pohybují po přímkách ve 2D prostoru, což se pro začátečníka jevilo jako mnohem lepší (jednodušší) možnost. Nesmírnou nevýhodou bohužel bylo, že zhruba polovina dosavadní práce tak musela být zahozena.

2.2.5 Build 0.1.0

texttexttext

2.3 Algoritmus AI počítačových protivníků

Při tvorbě AI nepřátel jsem se podobně jako jinde inspiroval hrami podobného žánru, tj. „top-down shoot-em-up“ hry, konkrétně podžánru „bullet hell“.
Tyto hry mají několik identifikujících znaků. Rozeberme si proto jednotlivé pojmy: „Top-down“, doslovně „zezhora dolů“, značí hru, ve které je kamera fixně umístěna nad herním polem, a ve které se hráč pohybuje po obrazovce (typicky po osách X a Y, ne však do hloubky) nezávisle.
„Shoot-em-up“ je takový typ hry, která obsahuje velké množství v poměru s hráčem velmi slabých protivníků. Z toho vyplývá, že hráč za určitý čas porazí větší množství protivníků, než je tomu u jiných žánrů. Shoot-em-up hry jsou proto často vnímány jako adrenalinové, a byly velmi populární v raných letech videoher na arkádových automatech. Jednou z prvních takových her byl např. *Galaxian*[1] od společnosti Bandai-Namco z roku 1979, nebo proslulí *Space Invaders*[2] z předcházejícího roku, kteří jsou často považováni za zakladatele žánru, mimo jiná prvenství[3].
Bullet hell je podžánr shoot-em-up her, který se vyznačuje tím, že je obrazovka „zaplavena“ nepřátelskými projektily, které sice nepůsobí valné poškození, ale jejich nebezpečí se skýtá v jejich množství. Hráč tak musí obratně manévrovat, aby se vyhnul co největšímu množství střel. Velmi častou vlastností je menší kolizní rámec projektilů, než jaká je jejich velikost na obrazovce. To umožňuje hráči provádět smělejší manévry a pomáhá balancovat obtížnost ve prospěch hráče.
Při tvorbě jsem se samozřejmě inspiroval i současnými tituly, mezi nimiž byla nejdůležitější *NieR: Automata*[4] od Platinum Games, resp. její top-down části.

Jak jsem se tedy pokusil docílit algoritmu, které by splňoval tyto žánrové charakteristiky.

[1] ATARI CORPORATION. *Galaxian* [online]. 1979 [cit. 2023-03-07]. Dostupné z: `https://www.mobygames.com/game/137/galaxian/`.

[2] ATARI CORPORATION. *Space Invaders* [online]. 1978 [cit. 2023-03-07]. Dostupné z: `https://www.mobygames.com/game/8806/space-invaders/`.

[3] Této hře je také přisuzován koncept více životů a udržování žebříčku nejvyšších skóre

[4] SQUARE ENIX et al. *NieR: Automata Game of the YoRHa Edition* [online]. 2017 [cit. 2023-03-07]. Dostupné z: `https://store.steampowered.com/app/524220/NieRAutomata/`.

Za prvé jsem jej rozdělil do dvou, na sobě minimálně závislých systémů. Jeden, který se stará o nepřátele jako celek (hlídá, aby ve hře nebylo příliš mnoho protivníků na jednou apod.) a druhý, který ovládá samotné nepřátele jednotlivě. Nejdříve se budu zabývat prvním zmíněným.

2.3.1 MasterSpawner

Tento Actor se stará převážně o to, aby počet nepřátel na poli nepřesáhl určitou hranici (která je stanovena úrovní obtížnosti). Je vyobrazen na obrázku v příloze číslo 2.
Nejprve ale krátké shrnutí základů vizuálního programování Blueprint. Program je rozdělen na jednotlivé nodes, z nichž některé obsahují příkazy, jiné operace či proměnné. Nodes s příkazy s bílým vstupem a výstupem exec se vyhodnocují zleva doprava, podle toho jak jsou propojeny. Ostatní nodes bez vstupu exec se vyhodnocují v ten okamžik, kdy je vyhodnocen node, do kterého vede jejich výstup.
Na začátku hry tento blueprint nejdříve nastaví některé důležité proměnné, a následně na každém z Actorů EnemySpawner zavolá funkci SpawnEnemy.

```
EnemyCount = 0
Spawners = [...]
#contains object references to all actors of class 'EnemySpawner'

for item in Spawners:
  item.SpawnEnemy(CombatStyle="ShootThreeRoundBurst")
  EnemyCount += 1
```

Na konci této setup fáze aktivuje node *Gate* v hlavním těle tohoto blueprintu.
Node *Gate* umožňuje projít signálu pouze v takovém případě, že se nachází ve stavu *Open*.
Pokud se toto stane, následují dvě věci. Zaprvé, pokud je současný počet nepřátel nižší než maximální dovolený, zavolá funkci Spawn v náhodném spawneru ve hře, s náhodnou hodnotou argumentu *CombatStyle*.
Za druhé přičte 1 k počtu nepřátel ve hře, a pokud je nový počet roven maximálnímu počtu, zavře *Gate*.
Zavřený *Gate* se otevře v takovém případě, že dojde ke smrti některého z nepřátel. Tento princip je shrnutý v následujícím pseudokódu:

```
EnemyCount = 0
MaximumEnemyCount = 0
GateIsOpen = True
Spawners = [...]

CombatStyles = [
  ShootContinuous,
  ShootThreeRoundBurst,
  ShootSeeking,
]

def main():
  if GateIsOpen:
    if EnemyCount < MaximumEnemyCount:
      spawner = Spawners[random]
      spawner.SpawnEnemy(CombatStyles[random])

      EnemyCount += 1

      if EnemyCount >= MaximumEnemyCount:
        GateIsOpen = False
    else:
      GateIsOpen = False

def EnemyDied():
  # called from outside the blueprint,
  # by the enemy that has died
  EnemyCount -= 1
  wait(randint(2,5))
  GateIsOpen = True

while True:
  main() #called every tick, ie. every new frame
```

Část II

Uživatelská příručka

1 Cíl hry

2 Ovládání

Závěr

Lorem ipsum

Resumé

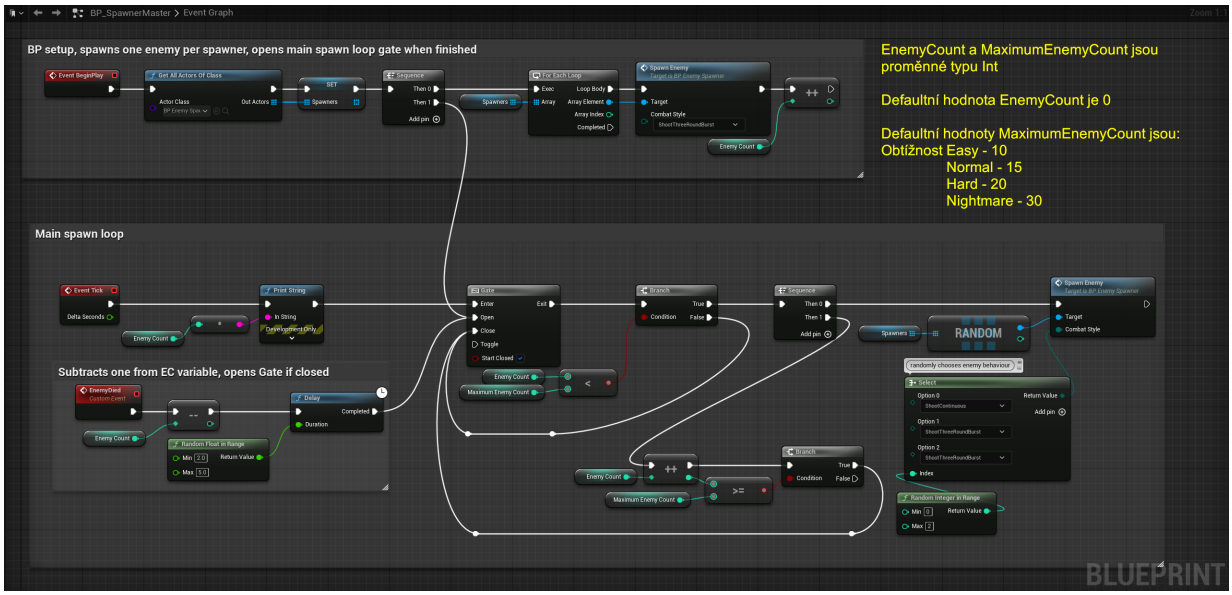
Lorem ipsum

Lorem ipsum ale anglicky

Příloha

Pozn.: Pokud není uveden zdroj, média jsou vlastní tvorby nebo ve veřejné doméně.

Obrázek 1: Snímek z verze 0.0.4



Obrázek 2: *BP_MasterSpawner*, kód pro kontrolu spawnování nepřátel

Seznam zdrojů a referencí

ATARI CORPORATION. *Galaxian* [online]. 1979 [cit. 2023-03-07]. Dostupné z: https://www.mobygames.com/game/137/galaxian/.

ATARI CORPORATION. *Space Invaders* [online]. 1978 [cit. 2023-03-07]. Dostupné z: https://www.mobygames.com/game/8806/space-invaders/.

SQUARE ENIX; PLATINUM GAMES. *NieR: Automata Game of the YoRHa Edition* [online]. 2017 [cit. 2023-03-07]. Dostupné z: https://store.steampowered.com/app/524220/NieRAutomata/.